INFORMATION PROCESSING SYSTEM AND STORAGE MEDIUM

Patent Number: JP2002073345
Publication date: 2002-03-12
Inventor(s): ETO TAKESHI
Applicant(s): FUJITSU LTD
Requested Patent: JP2002073345
Application Number: JP20000255317 20000825
Priority Number(s):
IPC Classification: G06F9/45
EC Classification:
Equivalents:

Abstract

PROBLEM TO BE SOLVED: To execute a program for a virtual machine faster.
SOLUTION: The program loader 16 reads out a byte code from the cluster file 12, decides whether a load location of this byte code and an argument setup byte code is the Java virtual machine 13 or the JIT compiler 15 based on content of a determination information storage unit 17 if the byte code is a subroutine call and iterates a loading process for the byte code to the decided load location. If the byte code is not the subroutine call, the loading location is decided to Java virtual machine 13. The content of determination information storage unit 17 is decided by the static analyzer 18 to analyze the content of the cluster file 16 or a person before the cluster file 12 is executed.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－73345
（P2002－73345A）
(43)公開日　平成14年３月12日(2002.3.12)

(51)Int.Cl.⁷	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　9/45		Ｇ０６Ｆ　9/44	３２０Ｃ　５Ｂ０８１

審査請求　未請求　請求項の数５　ＯＬ　（全７頁）

(21)出願番号　特願2000－255317(P2000－255317)

(22)出願日　平成12年８月25日(2000.8.25)

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　衛藤　健
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

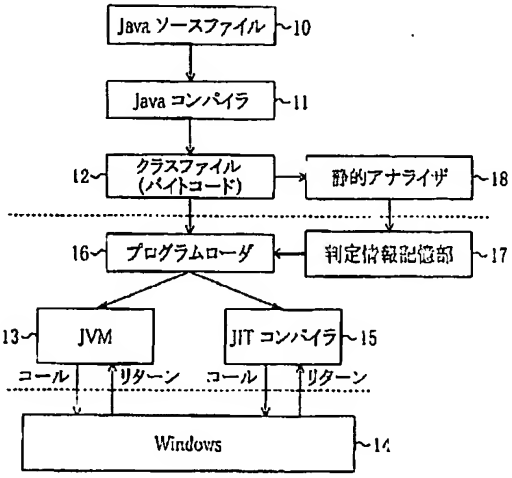
(74)代理人　100092587
弁理士　松本　貞吉

Ｆターム(参考)　5B081 AA09 DD01

(54)【発明の名称】　情報処理装置及び記録媒体

(57)【要約】
【課題】仮想マシンのプログラムをより高速に実行する。
【解決手段】プログラムローダ１６は、クラスタファイル１２からバイトコードを読み出し、バイトコードがサブルーチンコールであれば、判定情報記憶部１７の内容に基づいてこのバイトコード及び引数設定バイトコードのロード先を、Ｊａｖａ仮想マシン１３にするかＪＩＴコンパイラ１５にするかを決定し、決定されたロード先にバイトコードをロードするという処理を繰り返し実行する。バイトコードがサブルーチンコールでなければロード先をＪａｖａ仮想マシン１３にする。判定情報記憶部１７の内容は、クラスファイル１２の実行前に、クラスタファイル１６の内容を解析する静的アナライザ１８又は人により決定される。

本発明の一実施形態のJavaプログラム実行環境を示すソフトウェア概略ブロック図



【特許請求の範囲】

【請求項１】　ＯＳプラットフォーム上で仮想マシンがその仮想マシン命令コードを解釈実行する情報処理装置において、

仮想マシン命令コード列が格納されるプログラム記憶部と、

判定情報が格納される判定情報記憶部と、

ロードされた仮想マシン命令コードをコンパイルして実行するＪＩＴコンパイラがインストールされた記憶部と、

該プログラム記憶部から仮想マシン命令コードを読み出し、該判定情報に基づいてこの仮想マシン命令コードのロード先を該仮想マシンにするか該ＪＩＴコンパイラにするかを決定し、決定されたロード先に該仮想マシン命令コードをロードするという処理を繰り返し実行するプログラムローダがインストールされた記憶部と、

を有することを特徴とする情報処理装置。

【請求項２】　上記判定情報は、参照命令コードを有し、上記プログラムローダが動作する前に上記判定情報記憶部に格納されており、

上記プログラムローダは、上記プログラム記憶部から読み出した仮想マシン命令コードが該参照命令コードと一致すると判定すれば、該仮想マシン命令コードを上記ＪＩＴコンパイラにロードする、ことを特徴とする請求項１記載の情報処理装置。

【請求項３】　上記判定情報は、比較値とカウントとをさらに有し、

上記プログラムローダは、上記プログラム記憶部から読み出した仮想マシン命令コードが該参照命令コードと一致すると判定すれば、該カウントを変化させ、該カウントと該比較値との比較結果に基づいて該仮想マシン命令コードの上記ロード先を判定する、

ことを特徴とする請求項２記載の情報処理装置。

【請求項４】　上記プログラムローダが動作する前に、上記仮想マシン命令コード列を解析して上記参照命令コードを決定する静的アナライザがインストールされた記憶部、

をさらに有することを特徴とする請求項２又は３記載の情報処理装置。

【請求項５】　仮想マシン命令コードを読み出し、判定情報に基づいてこの仮想マシン命令コードのロード先を、仮想マシン命令コードをＯＳプラットフォーム上で解釈実行する仮想マシンにするか、仮想マシン命令コードをＯＳプラットフォーム上で動的にコンパイルし実行するＪＩＴコンパイラにするかを決定し、決定されたロード先に該仮想マシン命令コードをロードする、

という処理を繰り返し実行するプログラムローダが記録されていることを特徴とするコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ＯＳプラットフォーム上で、仮想マシンがその仮想マシン命令コードを解釈実行しＪＩＴコンパイラが仮想マシン命令コードを動的にコンパイルし実行する情報処理装置及びその処理を行うプログラムが記録されたコンピュータ読み取り可能な記録媒体に関する。

【０００２】

【従来の技術】図７は、Ｊａｖａ（登録商標）プログラム実行環境を示すソフトウェア概略ブロック図である。

【０００３】プログラムとデータを含むＪａｖａソースファイル１０は、Ｊａｖａコンパイラ１１により、バイトコード列とデータを含むクラスファイル１２に変換される。バイトコードは、ＪＶＭ（Java Virtual Machine）１３により解釈され実行される。ＪＶＭ１３は、Ｗｉｎｄｏｗｓ（登録商標）１４上で動作しており、Ｗｉｎｄｏｗｓ１４が持っている各種サブルーチンを利用することができる。ＪＶＭ１３を備えることにより、マシンに依存しないバイトコードを実行することができるという利点がある反面、オーバヘッドが大きいという問題がある。

【０００４】この問題を解決するために、Ｗｉｎｄｏｗｓ１４上で動作するＪＩＴ（Just-In-Time）コンパイラ１５を備え、クラスファイル１２のバイトコードをＪＩＴコンパイラ１５により実行時にコンパイル、すなわち動的にコンパイルし、その結果を記憶しておき、２回目以降このバイトコード（仮想マシンコード）を実行する場合には、再度コンパイルせずにコンパイルされたネイティブコード（ネイティブマシンコード）を実行するという方法が用いられている。

【０００５】

【発明が解決しようとする課題】しかしながら、クラスファイル１２の内容によっては、ＪＩＴコンパイラ１５で処理した方がＪＶＭ１３で処理するよりも遅くなる場合がある。この点は、Ｗｉｎｄｏｗｓ１４以外のＯＳ上のＪａｖａプラットホーム（Ｊａｖａコンパイラ１１、ＪＶＭ１３及びＪＩＴコンパイラ１５は、Ｗｉｎｄｏｗｓ１４上で動作するＪａｖａプラットホームを構成している。）で動作するＪａｖａプログラムについても同様であり、また、Ｓｍａｌｌｔａｌｋなどの他の仮想マシンプログラムについても同様である。

【０００６】本発明の目的は、このような問題点に鑑み、仮想マシンのプログラムをより高速に実行することが可能な情報処理装置及びその処理を行うプログラムが記録されたコンピュータ読み取り可能な記録媒体を提供することにある。

【０００７】

【課題を解決するための手段及びその作用効果】本発明では、ＯＳプラットフォーム上で仮想マシンがその仮想マシン命令コードを解釈実行する情報処理装置におい

て、仮想マシン命令コード列が格納されるプログラム記憶部と、判定情報が格納される判定情報記憶部と、ロードされた仮想マシン命令コードをコンパイルして実行するＪＩＴコンパイラがインストールされた記憶部と、該プログラム記憶部から仮想マシン命令コードを読み出し、該判定情報に基づいてこの仮想マシン命令コードのロード先を該仮想マシンにするか該ＪＩＴコンパイラにするかを決定し、決定されたロード先に該仮想マシン命令コードをロードするという処理を繰り返し実行するプログラムローダがインストールされた記憶部とを有する。

【０００８】この情報処理装置によれば、１つ以上の仮想マシン命令コードが読み出される毎に、判定情報の内容に応じて仮想マシン又はＪＩＴコンパイラで実行されるので、仮想マシンのみ又はＪＩＴコンパイラのみで仮想マシン命令コードを実行する場合よりも処理時間が短縮される。

【０００９】本発明の他の目的、構成及び効果は以下の説明から明らかになる。

【００１０】

【発明の実施の形態】以下、図面を参照して本発明の一実施形態を説明する。

【００１１】図１は、本発明の一実施形態のＪａｖａプログラム実行環境を示すソフトウェア概略ブロック図である。

【００１２】このソフトウェアは、コンピュータや携帯電話などの情報処理装置にインストールされている。

【００１３】本実施形態では、クラスファイル１２のバイトコードを、その内容に応じてＪＶＭ１３で解釈実行し又はＪＩＴコンパイラ１５で動的にコンパイルし実行するという処理を繰り返し行うことにより、ＪＶＭ１３のみ又はＪＩＴコンパイラ１５のみで実行する場合よりも高速化を図っている。

【００１４】バイトコードのロード先をＪＶＭ１３とするかＪＩＴコンパイラ１５とするかは、プログラムローダ１６により、判定情報記憶部１７の内容に基づいて判定される。判定情報記憶部１７の内容は、クラスファイル１２の実行前に、クラスファイル１２の内容を解析する静的アナライザ１８又は人（プログラム開発者やシステム運用者など）により決定される。

【００１５】図２は、図１のクラスファイル１２の実行手順を示す概略フローチャートである。

【００１６】（Ｓ１）プログラムローダ１６は、クラスファイル１２から次のバイトコードを読み出す。

【００１７】（Ｓ２）プログラムローダ１６は、このコードが終了コードであれば処理を終了し、そうでなければ次のステップＳ３へ進む。

【００１８】（Ｓ３）このバイトコードがサブルーチンコールでなければステップＳ４へ進む。このバイトコードがサブルーチンコールであれば、プログラムローダ１６は、クラスファイル１２からこれに関連したバイトコード群（現在のバイトコードで呼び出されるサブルーチン）をクラスファイル１２から読み出し、バイトコード群が、過去にＪＩＴコンパイラ１５でコンパイル済みでありその結果がメモリに格納されていればステップＳ７へ進み、そうでなければステップＳ４へ進む。

【００１９】（Ｓ４）ステップＳ１で読み出したバイトコードがサブルーチンコールでなければステップＳ５へ進む。このバイトコードがサブルーチンコールであれば、プログラムローダ１６は、判定情報記憶部１７の内容を参照し、バイトコード群をコンパイルするかどうかを後述のようにして判定する。コンパイルしないと判定した場合にはステップＳ５へ進み、すると判定した場合にはステップＳ６へ進む。

【００２０】（Ｓ５）プログラムローダ１６は、バイトコードをＪＶＭ１３にロードする。ＪＶＭ１３は、このバイトコードを解釈し、得られたネイティブコードをＭＰＵに供給して実行させる。次にステップＳ１へ戻る。

【００２１】（Ｓ６）プログラムローダ１６は、バイトコード群をＪＩＴコンパイラ１５にロードする。ＪＩＴコンパイラ１５は、このバイトコード群をコンパイル（ネイティブコードに変換）する。

【００２２】（Ｓ７）ＪＩＴコンパイラ１５は、変換されているネイティブコードをＭＰＵに供給して実行させる。次に、ステップＳ１へ戻る。

【００２３】図１において、ＪＶＭ１３及びＪＩＴコンパイラ１５はいずれもＷｉｎｄｏｗｓ１４上で動作し、Ｗｉｎｄｏｗｓ１４が持っているサブルーチンを呼び出すことによりこれを利用することができる。

【００２４】図３は、クラスファイル１２のバイトコードを実行中にＷｉｎｄｏｗｓ１４のグラフィック描画ライブラリが呼び出され、その実行中にさらにデバイスドライバが読み出される場合を示している。このような処理では、グラフィック描画ライブラリ及びデバイスドライバでの処理時間が比較的長いので、グラフィック描画ライブラリを呼び出すバイトコード群をＪＶＭ１３で実行してもＪＩＴコンパイラ１５で実行してもほぼ同じになる。この場合、ＪＩＴコンパイラ１５で実行すると、コンパイルされたネイティブコード列を記憶しておく必要があるので、プログラムローダ１６はこのバイトコード群をＪＶＭ１３にロードする。

【００２５】図４の（Ａ）及び（Ｂ）はいずれも、あるバイトコード群がＪＶＭ１３で繰り返し実行される場合とＪＩＴコンパイラ１５で繰り返し実行される場合の処理時間の比較を示す。

【００２６】ＪＶＭ１３で実行する場合には、毎回同じ処理時間ｔ１となるが、ＪＩＴコンパイラ１５で実行する場合にはバイトコード群が１回コンパイルされると２回目以降はコンパイルが省略されるので、２回目以降の処理時間ｔ３は１回目の処理時間（ｔ２＋ｔ３）よりも

コンパイル時間ｔ２だけ短くなる。なお、時間ｔ３にもオーバヘッドが含まれている。

【００２７】図４（Ａ）では、あるバイトコード群を１回のみ実行する場合には、ＪＶＭ１３を用いた方が高速であるが、このバイトコード群を２回実行する場合にはＪＶＭ１３とＪＩＴコンパイラ１５のいずれを用いてもほぼ同じとなり、このバイトコード群を３回以上実行する場合にはＪＩＴコンパイラ１５を用いた方が処理時間を短縮することができることを示している。また、図４（Ｂ）は、あるバイトコード群を１回のみ実行する場合にはＪＶＭ１３とＪＩＴコンパイラ１５のいずれを用いてもほぼ同じ時間となるが、このバイトコード群を２回以上実行する場合には、ＪＩＴコンパイラ１５を用いた方が処理時間を短縮できることを示している。

【００２８】このような解析が、図１の静的アナライザ１８又は人により行われて、判定情報記憶部１７に格納すべき情報が決定される。

【００２９】図５は、判定情報記憶部１７の内容の具体例と、これに関係した部分とを示す説明図である。

【００３０】判定情報記憶部１７には、ＪＩＴコンパイラ１５で実行される可能性のあるサブルーチンコールバイトコードＢＣα～ＢＣδ（引数を設定するバイトコードは含まれない。）が参照バイトコードとして予め格納されている。各参照バイトコードには、後述のフラグＦ、比較値Ｎ及びカウントｋの情報が付加されている。フラクＦ及び比較値Ｎは、参照バイトコードと同様に、静的アナライザ１８又は人により決定される。

【００３１】図６は、クラスファイル１２から読み出された１つのサブルーチンコールバイトコードＢＣｉに対する図５のプログラムローダ１６の処理を示すフローチャートである。

【００３２】（Ｓ９）判定情報記憶部１７内の各参照バイトコードに対応したカウントｋをゼロクリアする。

【００３３】（Ｓ１０）バイトコードＢＣｉを判定情報記憶部１７の次の参照コードと比較する。最初は、バイトコードＢＣｉを参照コードＢＣαと比較する。

【００３４】（Ｓ１１）判定情報記憶部１７に次の参照バイトコードがなければステップＳ１２へ進み、そうでなければステップＳ１３へ進む。

【００３５】（Ｓ１２）バイトコードＢＣｉをＪＶＭ１３にロードし、処理を終了する。

【００３６】（Ｓ１３）比較結果が一致すればステップＳ１４へ進み、そうでなければステップＳ１０へ戻る。

【００３７】（Ｓ１４）Ｆ＝‘０’であればステップＳ１５へ進み、Ｆ＝‘１’であればステップＳ１８へ進む。

【００３８】（Ｓ１５）カウントｋを１だけインクリメントする。

【００３９】（Ｓ１６）Ｎ＜ｋであればステップＳ１０へ戻り、Ｎ＝ｋであれば次のステップＳ１７へ進む。

【００４０】（Ｓ１７）フラグＦを‘１’にする。

【００４１】（Ｓ１８）バイトコードＢＣｉに関連したバイトコードをクラスファイル１２から読み出し、これとバイトコードＢＣｉとからなるバイトコード群をＪＩＴコンパイラ１５にロードする。

【００４２】このような処理により、バイトコードＢＣｉとＦ＝‘１’の参照バイトコードとが一致する場合にはバイトコードＢＣｉを含むバイトコード群がＪＩＴコンパイラ１５で実行され、バイトコードＢＣｉとＦ＝‘０’の参照バイトコードとが一致する場合にはこのバイトコードＢＣｉがＮ回以上出現したときにバイトコードＢＣｉを含むバイトコード群がＪＩＴコンパイラ１５で実行される。例えば、図４（Ａ）の場合にはＦ＝‘１’、Ｎ＝２であり、バイトコードＢＣｉが２回以上出現したときに、その後３回以上出現すると予測して、バイトコードＢＣｉを含むバイトコード群をＪＩＴコンパイラ１５で実行する。図４（Ｂ）の場合にはＦ＝‘０’であり、バイトコードＢＣｉが出現すればこれを含むバイトコード群を必ずＪＩＴコンパイラ１５で実行する。

【００４３】本実施形態によれば、バイトコードが上述のようにその内容に応じてＪＶＭ１３又はＪＩＴコンパイラ１５で実行されるので、ＪＶＭ１３のみ又はＪＩＴコンパイラ１５のみでバイトコードを実行する場合よりも処理時間が短縮される。

【００４４】なお、本発明には外にも種々の変形例が含まれる。

【００４５】例えば、バイトコード群をＪＩＴコンパイラ１５で実行した場合の１回目の時間と２回目の時間ｔ２及びｔ３を予め判定情報記憶部１７に格納しておき、クラスファイル１２の実行中において、このバイトコード群が最初に出現したときプログラムローダ１６はこれをＪＶＭ１３にロードし、その実行時間ｔ１を測定してこれを判定情報記憶部１７に格納し、ｔ１～ｔ３の値に基づいて比較値Ｎを決定するようにしてもよい。

【００４６】図５において、フラグＦを用いずに、比較値Ｎとカウントｋとを用い、ｋの初期値を０とし、Ｆ＝‘１’に対応したＮの値を１とし、バイトコードＢＣｉと参照バイトコードとが一致した場合にｋを１だけインクリメントし、ｋ＝Ｎのときバイトコード群をＪＩＴコンパイラ１５にロードするようにしてもよい。また、ｋの初期値をＮとし、Ｆ＝‘１’に対応したＮの値を１とし、バイトコードＢＣｉと参照バイトコードとが一致した場合にｋを１だけデクリメントし、ｋ＝０のときバイトコード群をＪＩＴコンパイラ１５にロードするようにしてもよい。

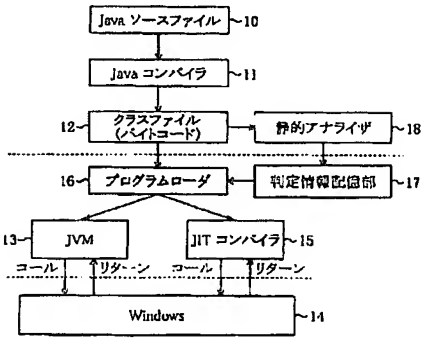
【００４７】本発明はＪａｖａ言語以外の仮想マシンにも適用される。仮想マシンは、Ｗｉｎｄｏｗｓ１４以外のＯＳ上で動作するものであってもよい。

【図面の簡単な説明】

【図１】本発明の一実施形態のＪａｖａプログラム実行環境を示すソフトウェア概略ブロック図である。
【図２】図１のクラスファイルの実行手順を示す概略フローチャートである。
【図３】クラスファイルのバイトコードを実行中にＷｉｎｄｏｗｓのグラフィック描画ライブラリが呼び出され、その実行中にさらにデバイスドライバが読み出される場合を示す説明図である。
【図４】（Ａ）及び（Ｂ）はいずれも、あるバイトコード群がＪＶＭで繰り返し実行される場合とＪＩＴコンパイラで繰り返し実行される場合の処理時間の比較を示す線図である。
【図５】図１の判定情報記憶部の内容の具体例と、これに関係した部分とを示す説明図である。
【図６】クラスファイルから読み出された１つのサブルーチンコールバイトコードＢＣｉに対する図５のプログラムローダの処理を示すフローチャートである。
【図７】従来のＪａｖａプログラム実行環境を示すソフトウェア概略ブロック図である。
【符号の説明】
１０　Ｊａｖａソースプログラム
１１　Ｊａｖａコンパイラ
１２　クラスファイル
１３　ＪＶＭ
１４　Ｗｉｎｄｏｗｓ
１５　ＪＩＴコンパイラ
１６　プログラムローダ
１７　判定情報
１８　静的アナライザ
Ｆ　フラグ
Ｎ　比較値
ｋ　カウント
ＢＣｉ　バイトコード

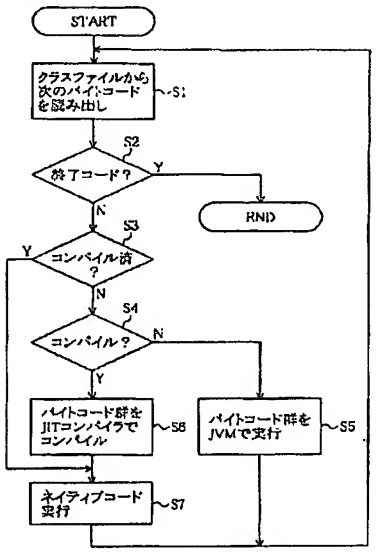
【図１】

本発明の一実施形態のJavaプログラム実行環境を示すソフトウェア概略ブロック図



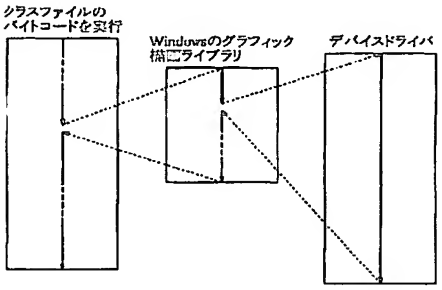
【図２】

図1のクラスファイルの実行手順を示す概略フローチャート



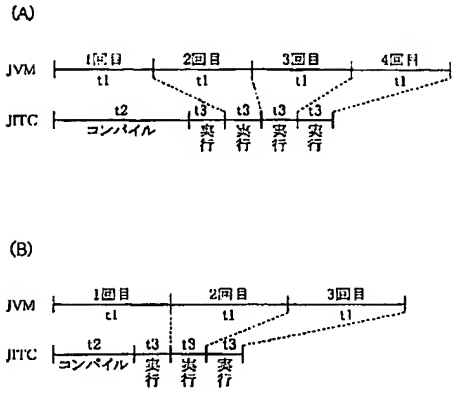
【図３】

クラスファイルのバイトコードを実行中にWindowsのグラフィック描画ライブラリが呼び出され、その実行中にさらにデバイスドライバが読み出される場合を示す説明図



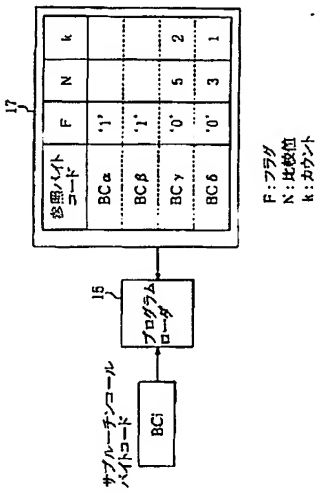
【図４】

(A)及び(B)はいずれも、あるバイトコード群がJVMで繰り返し実行される場合とJITコンパイラで繰り返し実行される場合の処理時間の比較を示す説図



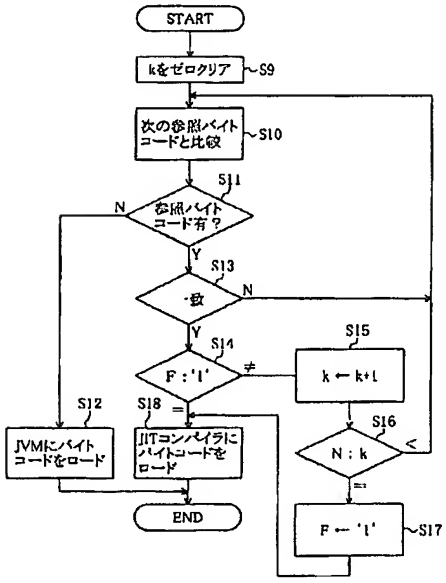
【図５】

図1の判定情報記憶部の内容の具体例と、これに関係した部分とを示す説明図



【図６】

クラスファイルから読み出された1つのサブルーチンコールバイトコードBCiに対する図5のプログラムローダの処理を示すフローチャート



【図７】

従来のJavaプログラム実行環境を示すソフトウェア構造ブロック図